Step-by-Step Guide to create a Static Web App in Microsoft Azure to Display “Student Count by Country”

Objective

The goal of this assignment is to create a static web application hosted on Azure that retrieves and displays the student count by country from data stored in an Azure SQL Database.

Steps to Complete the Project

---

1. Prepare Your Data and Environment. See the attached **Step-by-Step Guide to Building a Data Pipeline in Azure Data Factory**

1.1 Ensure Data Availability

Ensure that you have a table in your Azure SQL Database that contains student data. Here’s an example SQL schema and data:

```
SQL code (minimum columns)
-- Create Students table
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    Name NVARCHAR(100),
    Country NVARCHAR(100)
);
```

1.2 Develop a Web Application

Create a simple web application to display the student count by country. Use HTML, CSS, and JavaScript.

```
Example index.html:
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Count by Country</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
        }
        table {
            width: 60%;
            margin: 20px auto;
            border-collapse: collapse;
        }
        th, td {
            padding: 10px;
            border: 1px solid #ddd;
        }
        th {
            background-color: #f4f4f4;
        }
    </style>
</head>
<body>
    <h1>Student Count by Country</h1>
    <table id="studentTable">
        <thead>
            <tr>
                <th>Country</th>
                <th>Student Count</th>
            </tr>
        </thead>
        <tbody>
            <!-- Data will be populated here by JavaScript -->
        </tbody>
    </table>
    <script>
        async function fetchStudentData() {
            try {
                const response = await fetch('https://your-api-endpoint');
                const data = await response.json();

                const tableBody = document.querySelector('#studentTable tbody');
                tableBody.innerHTML = '';

                data.forEach(row => {
                    const tr = document.createElement('tr');
                    const countryTd = document.createElement('td');
                    const countTd = document.createElement('td');

                    countryTd.textContent = row.Country;
                    countTd.textContent = row.StudentCount;

                    tr.appendChild(countryTd);
                    tr.appendChild(countTd);
                    tableBody.appendChild(tr);
                });
            } catch (error) {
                console.error('Error fetching data:', error);
            }
        }

        fetchStudentData();
    </script>
</body>
</html>
```

2. Deploy the Web Application to Azure Static Web Apps

2.1 Create a Static Web App

1. **Log in to the Azure Portal**.
2. **Create a New Static Web App**:
    - Navigate to **Static Web Apps** and click **+ Create**.
    - Fill in the required details:
        - **Subscription**: Your Azure subscription.
        - **Resource Group**: Select or create a resource group.
        - **Name**: Provide a unique name for your static web app.

        - **Region**: Choose a region close to your users.
        - **Source**: Choose your source control (e.g., GitHub).
        - **Build Presets**: Choose **Custom** for a custom build setup.
    - Click **Review + Create**, then **Create**.

2.2 Configure Deployment

- **Link Your Repository**: If using GitHub or Azure DevOps, link the repository where your index.html is stored.
- **Configure Build**: If using custom build commands, specify them here.

2.3 Deploy Your Web Application

- Push your code to your repository. Azure Static Web Apps will automatically build and deploy your application.

3. Create an API to Fetch Data from Azure SQL Database

3.1 Create an Azure Function

1. **Create an Azure Function App**:
    - Go to **Function Apps** in the Azure Portal and click **+ Add**.
    - Configure the necessary settings and click **Create**.
2. **Create a New Function**:
    - In your Function App, click **+ Add** to create a new function.
    - Choose the **HTTP trigger** template.
3. **Implement the Function**:

```
Example code using JavaScript:
const { ConnectionPool } = require('mssql');

module.exports = async function (context, req) {
    const config = {
        user: process.env.SQL_USER,
        password: process.env.SQL_PASSWORD,
        server: process.env.SQL_SERVER,
        database: process.env.SQL_DATABASE,
        options: {
            encrypt: true
        }
    };

    const pool = new ConnectionPool(config);
    try {
        await pool.connect();
        const result = await pool.request()
            .query('SELECT Country, COUNT(*) AS StudentCount FROM
Students GROUP BY Country');

        context.res = {
            status: 200,
            body: result.recordset
        };
    } catch (err) {
        context.res = {
            status: 500,
            body: err.message
        };
    } finally {
        pool.close();
    }
};
```

4. **Configure Application Settings**:
    - Set environment variables for the connection string details in the Function App’s **Configuration** settings.
5. **Deploy the Function**:
    - Deploy your function from your local environment or using the portal.

4. Connect Web Application to API

- Update the fetch URL in your index.html with the URL of your Azure Function API.

5. Test Your Application

- Access your static web app URL and ensure that it correctly fetches and displays the student count by country.

6. Document Your Work

- **Provide Screenshots**:
    - Azure portal setups.
    - Pipeline creation.
    - SQL query results.
- **Write a Short Explanation**:
    - Describe the objective.
    - Tools used.
    - Implementation steps.
    - Results of the “Student Count by Country” query.

Conclusion

You have now created a static web app in Microsoft Azure to display the student count by country using data stored in Azure SQL Database. Ensure that all components are working as expected and provide the necessary documentation for your assignment.